Info Navigator: A visualization tool for document searching and browsing

Conference or Workshop Item

Info Navigator: A Visualization Tool for Document Searching and Browsing

Matthew Carey, Daniel C Heesch and Stefan M Rüger *

Abstract

We present a text document search engine with several new visualization front-ends that aid navigation through the set of documents returned by a query (short "returned documents"*). Our methods are based on identifying and selecting keywords on the fly. The choice of keywords depends not only on the frequency of their occurrence within returned documents but also on the specificity of their occurrence just within these returned documents. Keywords are subsequently used to obtain a sparse document representation and to compute document clusters using a variant of the buckshot algorithm. One of the visualization front-ends uses the sparse document representation to obtain keyword clusters. We make use of the clustering to group the documents returned from the search visually, and to label the groups with their most salient keywords. The different front-ends cater for different user needs. Two of them can be employed to browse cluster information as well as to drill down or up in clusters and refine the search using one or more of the suggested keywords.*

1. Introduction

Broad one or two-word searches in conventional search engines are often plagued by low precision, returning thousands of documents as their output. A common problem with this is that users may have to sift through much irrelevant material before finding pertinent documents.

We suggest using a standard full-text search engine, but in addition computing keywords from the set of returned documents. These keywords can assist the user in a variety of ways: i) informing about issues related to the query, ii) narrowing down the query with additional query term suggestions, iii) clustering and iv) displaying and labelling the returned-document clusters.

In Section 2, we review our method of generating keywords. In order to be computationally efficient we generate a list of candidate keywords for each document at index time. This list is available at query time without having to access the original documents. Three criteria are applied for keyword selection: they must be of general potential interest, be specific for the returned-document set and be of discriminative power within this set. In our document set of around 550,000 documents the query "computer" will produce keywords like "software", "UNIX", "IBM" or "users".

When clustering documents from the returned-document set, we make use of a sparse document representation using keyword histogram vectors. Not only does this alleviate the curse of dimensionality that comes with the otherwise popular word histogram representations but it also reduces the computing time significantly. Section 3 describes the technical components and interaction of the whole search engine with its Information Navigator front-end, while Section 4 details three graphical interfaces that make use of the keyword-document matrix. These three interfaces are alternative views of the traditional ranked-list representation.

Our thesis is that these graphical cluster-based representations shift the user's mental load from slower thought-intensive processes such as reading to faster perceptual processes such as pattern recognition in a visual display. In our opinion, the one-dimensional ranked-list metaphor is too restrictive when the returned-document set is large. Furthermore, in conventional search engines, the documents are ultimately ranked with the aim of ordering them according to relevance to the user. This appears to be overly ambitious as even advanced ranking algorithms cannot know whether the user prefers documents about "hardware" or "software" when the query simply was "computer". Again, a graphical cluster-based interface might help users find what they want.

2. Keywords and Clustering

2.1. The Curse of Dimensionality and Dynamically Computed Keywords

The natural features of text documents are words or phrases, and a document collection can contain millions of such distinct features. The often-used word histogram representation of documents consequently leads to very high dimensional vectors. The intrinsic problem with this kind of representations is that any two randomly picked vectors in a high-dimensional hypercube tend to have a constant distance from each other, no matter what the measure is. As a consequence, clustering algorithms that are based on document distances become unreliable. For a more detailed discussion about the curse of dimensionality see [19].

Even after applying feature reduction, the number of features remains large. In our clustering experiments with

*M Carey, D C Heesch, S M Rüger are with the Department of Computing, South Kensington Campus, Imperial College London, London SW7 2AZ, England, UK

548,948 documents[1], a *candidate keyword* had to appear in at least three documents and in no more that 33% of all documents. This resulted in a vocabulary of around 222,872 candidate keywords. In our system we store, at index time, around 200 candidate keywords per document. A set R of documents returned by a query may still have a candidate-keyword vocabulary of well over 10,000 different words. As an example, the four sets of top 500 documents returned by the queries "mafia," "Clinton," "cancer" and "computer" use a vocabulary of between 14,000 and 17,000 different candidate keywords.

Document clustering has attracted much interest in the recent decades, eg [20, 8, 29, 17], and much is known about the importance of feature reduction in general, eg [14] and, in particular, clustering [27]. However, little has been done so far to facilitate feature reduction for document clustering of query results, with the notable exception of [22]. In contrast to the latter paper, which uses a tf-idf weighting scheme, we suggest ranking the importance of each such candidate keyword j with a weight

$$w_j = \frac{r_j}{d_j} \cdot r_j \log(|R|/r_j), \tag{1}$$

where $|R|$ is the total number of documents in the set R of returned documents, r_j is the number of documents in R containing word j, and d_j is the number of documents in the whole document collection D containing j. The second factor prefers words with medium-valued matched-document frequencies, while the first factor prefers words that specifically occur in the matched documents. The highest-ranked words are meant to be related to the query. Indeed, we have "software", "IBM", "UNIX" etc as the top-ranked words when querying for "computer". This seems to be a powerful approach to restrict the features of the matched documents to the top k ranked words, which we will call the *keywords*. One important aspect is that the features are computed at query time. Hence, when the above query is refined to "computer hardware", a completely new set of keywords emerges automatically.

2.2. Document Representation

For each matched document i we create a k-dimensional vector v_i, whose j-th component v_{ij} is a function of the number of occurrences t_{ij} of the j-th ranked keyword in the document i:

$$v_{ij} = \log_2(1 + t_{ij}) \cdot \log(|D|/d_j) \tag{2}$$

This is a variation of the tf-idf weight that gives less stress to the term frequency t_{ij} (the number of occurrences of the jth ranked keyword in document i). We project the vector v_i onto the k-dimensional unit sphere obtaining a normalized vector u_i that represents the document i. We deem the scalar product of u_a and u_b (i.e. the cosine of the angle between vectors v_a and v_b) a sensible *semantic similarity measure* between two documents a and b in the document subset R returned by a query with respect to the complete document collection D.

u may be viewed as a document representation matrix where the row vector u_i is a k-dimensional representation of document i and u_{ij} is viewed as the importance of keyword j for document i. In particular, $u_{ij} = 0$ if and only if document i does not contain keyword j. The number of features k can be controlled by the experimenter, and our experiments using the TREC data of human relevance judgements have shown that $k \approx 10$ yields superior clustering results [19]. Note that even if only the top ten keywords are used for the clustering and document representation, we might still display more keywords on the screen to assist the user in his or her search.

2.3. Document Clustering

Post-retrieval document clustering has been well studied, eg [9, 1, 15, 10, 31]. We deploy a variant of the Buckshot algorithm [9]. Each cluster contains a certain number of document vectors and is represented by their normalized arithmetic mean, the so-called centroid vector. In the first phase, hierarchical clustering with complete linkage operates on the best-ranked 150 documents. This can be done in a fraction of one second CPU time. Hierarchical clustering has the advantage that one can either prescribe the number N of clusters or let the number of clusters be determined by demanding a certain minimum similarity within a cluster. Either way, once clusters within the top-ranked documents are identified, their N centroids can be computed and used as a seed for standard iterative clustering of the remaining documents. This algorithm scales linear with the number $|D|$ of documents and the number N of clusters. In our experience, one cycle of iterative clustering is not only adequate but also preserves the cluster structure given by the top-ranked documents. 1,000 documents can thus be clustered in well under one second on a 2 GHz PC.

3. System Overview and Architecture

The system consists of two major subsystems: the server side components that index the data, carry out queries and process the results, and the graphical user interface. In the rest of this section we explain the constituent parts of both.

The search engine consists of three major parts: the indexing program that feeds the search engine, the search engine itself and the result processing program that adds the interesting word data, clusters and *Sammon Maps* [21] to the returned-document set.

The search engine is powered by a slightly modified version of the *mg search engine* that accompanies the Managing Gigabytes book [30]. It can be set to perform ranked query searches where the output is a list of document references and a brief section of the text.

For indexing, the mg engine expects to run an application or a shell script that delivers a series of documents to it via standard output, delimiting each document. The user needs to

[1]TREC CDs vol3 and vol4, including articles of the Los Angeles Times, the Financial Times, the Federal Register, Congress Records and the Foreign Broadcast Information Service, see http://trec.nist.gov

devise a suitable program to carry out this task. Our indexing program, while descending through the various document directories and listing their contents to the mg engine, it keeps a record of each word used in all the documents and of its document frequency.

After the first pass the indexing program has a data file containing the document frequency of each word. This file is used to create an ordered file of candidate keywords, an index into that file and an index file of document frequencies. Candidate keywords are those that fall inside a statistical boundary of document frequencies (see Section 2.1). The indexing program then makes a second pass through the document data files, recording, for each document, the frequency of each candidate keyword it contains. The data is stored in a variable length record file and an index into that file is created.

To obtain keywords, the candidate keyword and document data encapsulated in the above auxiliary index files are used by the processing program to calculate the weight of each candidate keyword using the statistical formula (1). The 100 highest weighted candidate keywords in the result set are kept as keywords. A sparse matrix of the incidents of the keywords in each document is then created. This matrix is clustered, and a Sammon map is generated from the cluster centroids. The document numbers, document snippets, the keywords, the sparse matrix of keyword document incidents, the clusters of document numbers and the Sammon map of the clusters are all piped out to the calling program.

Once the documents are indexed, the mg search engine is able to respond to a query by computing efficiently a ranked list of the references to those documents that contain the query words. Note that at this point a standard full-text inverted document file is used for processing a query.

The display applet consists of a query interface and three visualizations on different tabbed panels that are serviced by the same data manager module. This module is the piece of code responsible for storing the data and supplying it to the individual visualization modules.

The applet communicates directly with a servlet on the server-side. In the case of a search query the servlet runs a shell script that calls instances of the mg search engine and the result-processing program. The result is piped back to the servlet, which then creates a serialized data object to be transmitted to the applet. The applet uses the data to set the contents of the data manager and updates the display of all three visualizations.

When the user asks for a document to be displayed, the servlet generates a request for mg and retrieves the full content of the specified document. The servlet then transmits it as a serialized vector of lines to the applet to be displayed in a separate browser window.

The system as it stands is a thick client model. It minimizes use of the network by putting much of the work in the applet. Once the applet is installed in the client's browser only a limited demand will be made on network resources, both at query time and on document retrieval.

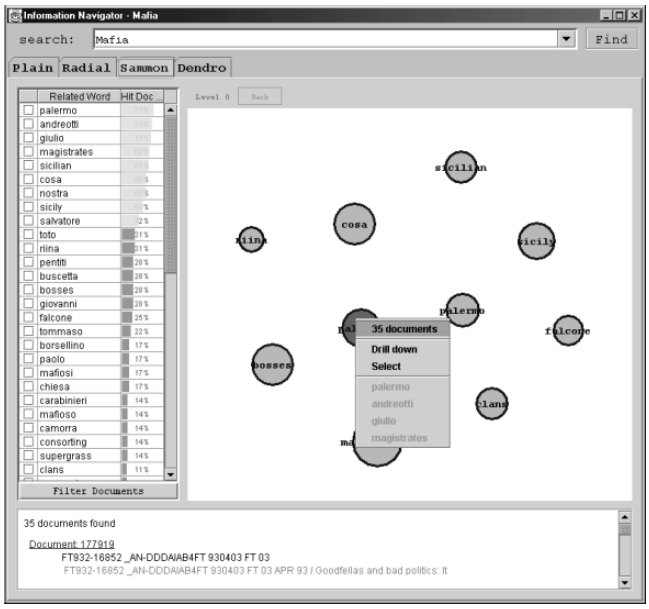

Figure 1. Sammon map for cluster-guided search

4. New Paradigms in Information Visualization

The last decade has witnessed an explosion in interest in the field of information visualization, e.g. [13, 7, 26, 11, 2, 6, 16, 3, 25, 33, 31, 32, 24, 5, 18]. We add three new techniques to the pool of existing visualization paradigms, based on our design study [4].

4.1. Sammon Cluster View

This paradigm uses a Sammon map to generate a two dimensional location from a many-dimensional vector of cluster centroids. This map is computed using an iterative gradient search [21]. The algorithm aims to represent n-dimensional points in a lower-dimensional space while attempting to preserve the pairwise distances between the objects. Clusters are thus arranged such that their mutual distances are indicative of their relationships in the n-dimensional space. The idea is to create a visual landscape for navigation.

The display has three panels, a scrolling table panel to the left, a graphic panel to the right and a scrolling text panel below (see Fig 1). In the graphic panel each cluster is represented by a circle and is labelled with its most frequent keyword. The radius of the circle informs about the size the cluster. The distance between any two circles in the graphic panel is an indication of the similarity of their respective clusters: the nearer the clusters, the more likely the documents contained with in will be similar. When the mouse passes over the cluster circle a 'tool tip' box in the form of a pop-up menu appears.

The first item in the cluster popup menu shows the count of documents in that cluster. Choosing this item displays a scrolled table of cluster keywords in the pane on the left-hand side of the visualization and a scrolled list of cluster document

hot-links and snippets appear in the scrolling text window at the bottom of the display.

The table of keywords includes a box field that can be selected. At the bottom of the table is a filter button that makes the scrolling text window display only the hot-links and snippets from documents that contain the selected keywords. The select item in the pop-up menu marks a cluster as selected and signals this with a flag. The other menu items serve to label the cluster with four significant keywords and are not selectable.

The drill down item in the pop-up menu performs a redisplay of the documents in the current cluster and all selected clusters (if any). Drill down in this sense pushes the current data manager onto a stack and creates a new data manager that consists of only the documents in the current and selected clusters. This new instance of the data manager re-clusters the subset of the original returned-document set and then creates a new Sammon map that in turn leads to a new display in this visualization. The level indication at the top of the display is incremented and the back button enabled. The back button pops the data manager from the stack and climbs up the hierarchy (drill up). The clustering algorithm used for reclustering on a drill down operation is the one described in 2.3 but this time implemented in the applet.

4.2. Dendro Map Visualization

The Dendro Map visualization represents documents as leaf nodes of a binary tree using the same clustering algorithm described earlier. With its plane-spanning property and progressive shortening of branches towards the periphery, the Dendro Map mimics the result of a non-Euclidean transformation of the plane as used in hyperbolic maps without suffering from the computational load typically associated with the latter. Owing to spatial constraints, the visualization depth is confined to five levels of the hierarchy with nodes of the lowest level representing either documents or subclusters. Different colours facilitate visual discrimination between individual documents and clusters. Next to each lowest level node is printed the most frequent keyword of the subcluster (or document). This forms a key component of the Dendro Map as it gives the user the cues needed for navigating through the tree. As the user moves the mouse pointer over an internal node, the colour of internal nodes and branches of the associated subcluster change colour from light blue to dark blue while the leaf nodes (i.e. document representations) turn bright red. As in the Sammon Map, a tool-tip window provides additional information about the cluster and can be used to display a table with a list of keywords associated with the cluster. The user may perform drill-down on any internal node. The selected node will as a result replace the current root node at the center and the entire display is re-organized around the new root. The multi-level approach of the Dendro Map allows the user to gain a quick overview over the document collection and to identify promising subsets.

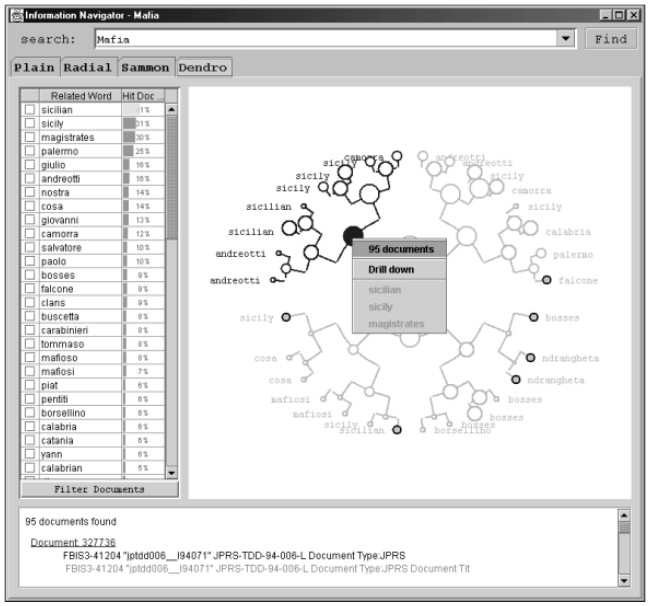

Figure 2. Dendro Map: a plane-spanning binary tree

4.3. Radial Interactive Visualization

Radial (Figure 3) is similar to VIBE [13], to Radviz [12] and to Lyberworld [11]. Radviz places the keyword nodes round a circle as dimensional anchors and the document nodes occur in the middle as if suspended by springs connecting them to keyword nodes. The greater the content, the stronger the springs' effect on the document's location. Hence, we make direct use of the representation matrix u without explicit clustering. Radial adds a level of user interaction to the metaphor introduced by RadViz and VIBE. Building on VIBE, Lyberworld takes a similar idea into three dimensions. In Lyberworld vector addition is used to position the documents nodes within a *relevance sphere*. The document's keyword content creates a vector between the centre axis of the sphere and the position of that keyword on the sphere's surface. The radius of the sphere is defined by the range of possible vector lengths. In Lyberworld the relevance sphere can be rotated so that the 2D computer display can give more of a clue as to the 3D location of the document node relative to the keyword nodes. Also the relative attractiveness of the keyword nodes can be enhanced to pull related documents towards them. Radial, staying with only two dimensions, dispenses with some of the perceptual complexity implicit in rendering a three dimensional model on a two dimensional screen.

Radial also uses the statistically collected keywords to differentiate the documents. Initially, the twelve highest ranking keywords are displayed in a circle. Any documents in the search set that contain those keywords are placed using a similar vector sum within the circle. As the mouse passes over the document nodes, a bubble displays a descriptive piece of text from the document. The dimensions of the circle are more

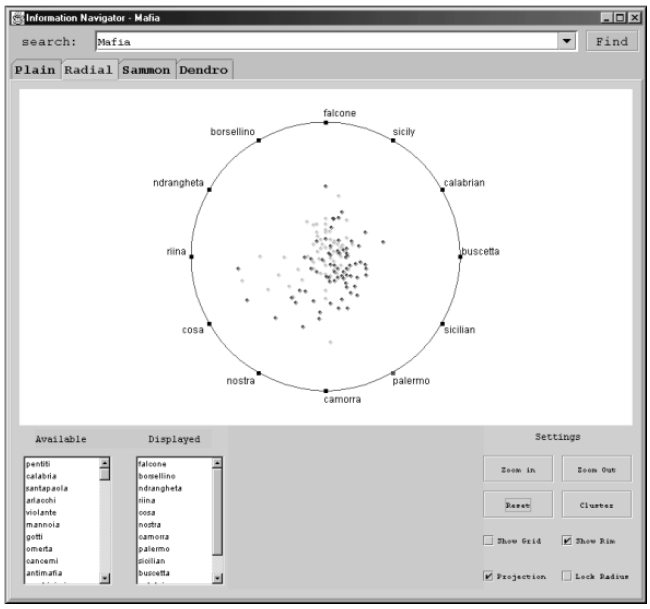

Figure 3. Radial visualization

arbitrary than in Lyberworld and if the display were simply based on a flat sum of vectors it would be possible for the document nodes to be outside the circle. However we have constrained their positions by projecting the radial locations through the arc-tangent function, with the result that the documents cannot be moved outside of the circle.

Dimensionality reduction has the effect that locations of document nodes become ambiguous. There may be many reasons for a node to have a particular position. To mitigate this ambiguity in Radial the user can click on a document node, and the keyword nodes that affect the location of document are highlighted. We believe that this is a novel and useful feature. Clues to the effects of different dimensions are also given when a keyword node is selected with the mouse: the document nodes which contain that keyword are highlighted. Similarly to Lyberworld the vector effect of a particular keyword node on the document set can be enhanced. However, in Radial it is achieved by grabbing the keyword node with the mouse and moving it away from the edge of the circle: All documents that contain this keyword follow the movement of the keyword, accordingly. Manual clustering can be effected by placing several keyword nodes together. Alternatively, a button allows the displayed keyword arrangement to be automatically clustered using the *columns* of the matrix u. Note that document clustering was done using the *rows* of this matrix.

The choice of keywords used in the display can be enhanced and reduced by clicking on two visible lists of words. Zoom buttons allow the degree of projection to be increased or reduced so as to distinguish between document nodes around the edges of the display or at the centre.

The Radial visualization appears to be a good interactive tool to structure the document set according to one's own preferences by shifting keywords around in the display. The shortcoming of the Radial visualization is that it can only say something about the documents in the result set that contain the particular, limited set of keywords which are displayed. When too many keywords are displayed the whole display becomes difficult to interpret. This is where the cluster-based visualizations (Dendro Map and Sammon Map) prove their merits.

4.4. Unified Approach

The application as a whole offers the possibility of browsing the same result set in several different ways simultaneously. The cluster-based visualizations give a broader overall picture of the result, while the Radial visualization allows the user to focus on subsets of keywords. Also, as the clusters are approximations that bring to the fore particular keywords, it may be useful to return to the Radial visualization and examine the effect of these keywords upon the whole document set. The Radial visualization will perhaps be more fruitful if the initial keywords match the user's area of interest. The Sammon Map will let the user dissect search sets and re-cluster subsets, gradually homing in on target sets. Again the user may have recourse to the Radial visualization and apply those keywords to the whole result set. In the end, he or she will need to read only a small fraction of the retrieved documents. The cluster-based visualizations give a visual analogy of the structure implicit in the library classification scheme. The Radial visualization throws up the effects of keywords that cause cross references between documents, and allows the user to skim between subject areas.

5. Evaluation

The literature of information visualization is still one mainly of construction, not of evaluation; for some exceptions to this rule see [28, 23]. Our evaluation is a three level approach. For level one, we performed experiments to assess the quality of the clustering process based on human-expert data, ie, the ability to separate relevant documents from irrelevant documents [19]. We used the 1997-1998 sub-collection of the TREC data with 528,155 documents, 100 queries and corresponding relevance assessments. The question we posed was whether our clustering algorithms would produce clusters in which the concentration of relevant documents was either very high or very low. The results showed a compelling evidence for the validity of the Clustering Hypothesis [27] for post-retrieval document sets and for the use of low-dimensional document representations using the keyword computation discussed in Section 2. This study gave the green light for developing designs for cluster visualizations. For level two, in the process of developing this software, we were able to get invaluable feedback from students. The application was posted on the web and we asked a group of users to complete a pre-use questionnaire. We then had them employ specific visualization interfaces to execute a collection of pre-set queries that were sufficiently obscure for the answers not to be known

beforehand, and yet to be in the data set somewhere. Next, we asked the users to apply the system to queries of their own choice and, finally, we asked them to complete a post-use questionnaire that contained both narrow questions about the application and more open areas for user comment. This second-level evaluation served to refine the design of the visualization interfaces.

The third level of evaluation is a proper and formal user study currently carried out through collaboration with the Ben-Gurion University of the Negev.

6. Conclusions

Our work has contributed to the visualization and browsing of the set of document returned by a search engine in a number of ways. 1) We can identify relevant features of this document set: the keywords. These are used for dimensionality reduction and improved clustering, cluster labelling, query refinement and visualization. 2) Using Sammon's algorithm we are able to create a setting with a holistic view giving primarily information about a first-order cluster structure and inter-cluster relations. The main purpose is to quickly weed out irrelevant clusters and drill down in one or more relevant clusters. 3) The distorted binary tree as used in our Dendro Map implementation allows us to visualize several levels of the cluster hierarchy and thus aids the user in quickly narrowing down her or his search to a small subset of documents. 4) A keyword clustering with the Radial visualization gives rise to another novel document clustering approach, one where the user can control the building of groups by interactively moving keywords, making it particularly useful for an experimental, user-driven approach to form clusters.

References

[1] R Allen, P Obry, and M Littman. An interface for navigating clustered document sets returned by queries. In *ACM Conf on Organizational Computing Systems*, 1993.

[2] M Ankerst, D Keim, and H Kriegel. Circle segments: A technique for visually exploring large multidimensional data sets. In *IEEE Visualization*, 1996.

[3] J Assa, D Cohen-Or, and T Milo. Displaying data in multidimensional relevance space with 2d visualization maps. In *IEEE Visualization*, 1997.

[4] P Au, M Carey, S Sewraz, Y Guo, and S Rüger. New paradigms in information visualisation. In *SIGIR*, 2000.

[5] K Börner. Visible threads: A smart VR interface to digital libraries. In *SPIE*, 2000.

[6] S Card. Visualizing retrieved information: A survey. *IEEE Computer Graphics and Applications*, 16(2), 1996.

[7] M Chalmers and P Chitson. Bead: Explorations in information visualisation. In *SIGIR*, 1992.

[8] B Croft. *Organizing and searching large files of documents*. PhD thesis, University of Cambridge, 1978.

[9] D Cutting, D Karger, J Pedersen, and J Tukey. Scatter/gather: a cluster-based approach to browsing large document collections. In *SIGIR*, 1992.

[10] M Hearst and J Pedersen. Reexamining the cluster hypothesis: scatter/gather on retrieval results. In *SIGIR*, 1996.

[11] M Hemmje, C Kunkel, and A Willet. Lyberworld - a visualization user interface supporting fulltext retrieval. In *SIGIR*, 1994.

[12] P Hoffman, G Grinstein, and D Pinkney. Dimensional anchors: A graphic primitive for multidimensional multivariate information visualizations. In *NPIVM 99*, 2000.

[13] R Korfhage. To see or not to see - is that the query? In *SIGIR*, 1991.

[14] P Krishnaiah and L Kanal, editors. *Handbook of Statistics*, volume 2. North-Holland, 1982.

[15] A Leouski and B Croft. An evaluation of techniques for clustering search results. Technical Report IR-76, Computer Science, UMass, Amherst, 1996.

[16] L Nowell, R France, D Hix, L Heath, and E Fox. Visualizing search results: Some alternatives to query-document similarity. In *SIGIR*, 1996.

[17] E Rasmussen. Clustering algorithms. In W Frakes and R Baeza-Yates, editors, *Information Retrieval: Data Structures and Algorithms*. Prentice Hall, 1992.

[18] P Rheingans and M desJardins. Visualizing high-dimensional predictive model quality. In *IEEE Visualization*, 2000.

[19] S Rüger and S Gauch. Feature reduction for document clustering and classification. Technical report, Computing Department, Imperial College London, UK, 2000.

[20] Gerard Salton. *Automatic information organization and retrieval*. McGraw-Hill, New York, 1968.

[21] J Sammon. A nonlinear mapping for data structure analysis. *IEEE Trans on Computers*, C-18(5), 1969.

[22] H Schütze and C Silverstein. Projections for efficient document clustering. In *SIGIR*, 1997.

[23] M Sebrechts, J Vasilakis, M Miller, J Cugini, and S Laskowski. Visualization of search results: A comparative evaluation of text, 2d and 3d interfaces. In *SIGIR*, 1999.

[24] C Shaw, J Kukla, I Soboroff, D Ebert, C Nicholas, A Zwa, E Miller, and D Roberts. Interactive volumetric information visualization for document corpus management. *Intl Journal on Digital Libraries*, 2(2/3), 1999.

[25] B Shneiderman, D Feldman, A Rose, and X Ferre' Grau. Visualizing digital library search results with categorical and hierarchical axes. In *ACM Digital Libraries*, 2000.

[26] A Spoerri. InfoCrystal: A visual tool for information retrieval & management. In *CIKM*, 1993.

[27] C van Rijsbergen. *Information Retrieval*. Butterworth, London, 2nd edition, 1979.

[28] A Veerasamy and R Heikes. Effectiveness of a graphical display of retrieval results. In *SIGIR*, 1997.

[29] E Voorhees. The cluster hypothesis revisited. In *SIGIR*, 1985.

[30] I Witten, A Moffat, and T Bell. *Managing Gigabytes*. Morgan Kaufmann, 1999.

[31] O Zamir and O Etzioni. Web document clustering: A feasibility demonstration. In *SIGIR*, 1998.

[32] O Zamir and O Etzioni. Grouper: A dynamic clustering interface to web search results. In *WWW-8*, 1999.

[33] M Zhou and S Feiner. Visual task characterization for automated visual discourse synthesis. In *CHI*, 1998.

Acknowledgements: This work was partially supported by the EPSRC, UK.